View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Purdue E-Pubs

Purdue University

Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1987

A Priori Grid Adaption Strategies for Elliptic PDEs

Calvin J. Ribbens

Report Number:
87-667

Ribbens, Calvin J., "A Priori Grid Adaption Strategies for Elliptic PDEs" (1987). *Department of Computer Science Technical Reports.* Paper 578.
https://docs.lib.purdue.edu/cstech/578

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

A PRIORI GRID ADAPTION STRATEGIES FOR ELLIPTIC PDES*

Calvin J. Ribbens†

CSD-TR 667
March 1987

Abstract

Adaptive grid techniques are often used to solve difficult PDEs, especially those which model time-dependent physical systems. The usual approach is to adapt the grid from one time step to the next, based on the solution at the previous time step. Grid adaption is less commonly applied to elliptic PDEs, which model the steady state behavior of a physical system. In this paper we examine whether *a priori* problem information may be used to efficiently generate adapted grids. We describe two grid adaption algorithms which use this approach. Both methods adapt tensor product grids by moving grid points. We illustrate the performance of these methods on a difficult problem. We describe a computational framework in which methods of this kind may be developed and researched, and which allows existing numerical methods to be used whenever possible.

1 Introduction

We examine the potential of *a priori* grid adaption methods for the numerical solution of partial differential equations. The problems we consider are two-dimensional, second-order, linear, elliptic boundary value problems. The general form of such a problem is

$$au_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + fu = g \quad \text{on } R, \tag{1}$$

$$pu_x + qu_y + ru = s \quad \text{on } \partial R, \tag{2}$$

where a, b, c, d, e, f, g, p, q, r and s are functions of x and y. We seek an approximation to the unknown function u which satisfies (1) in the two-dimensional region R and (2) on the boundary ∂R. While problems of the form (1) and (2) are the main focus of our work, we believe our approach to grid adaption may be applied to other classes of PDEs as well.

*To be presented at the IMACS International Symposium on Computer Methods for Partial Differential Equations, Lehigh University, June 23-26, 1987.

†Supported in part by National Science Foundation grant MS-8301589 and by Air Force Office of Scientific Research grant AFOSR 84-0385.

The power of adaption for improving the accuracy of grid based numerical methods is well-known (see [3] and [11] for example). In the most common approaches the grid is moved, or locally refined, based on a previous solution to the problem. In the case of time-dependent problems, a natural strategy is to base the adaption on the approximate solution at the most recent time-step. Thus the grid moves with the solution through time. Various criteria are used to drive the adaption. An estimate of the spatial discretization error is used in [1], [4] and [6]. The residual in the approximate solution is used in [7]. Another common aspect of grid adaption is that the computation of new grid locations or of areas where refinement is needed, is often relatively expensive. In [7] the position of each grid point at the next time step is included in the set of unknowns for the numerical method. A system of ordinary differential equations is solved in [2]. A system of elliptic PDEs may be solved to generate grids with certain adaptive characteristics [12].

For steady state problems a solution at a previous time step is not available. One would also like to avoid solving a second problem of equal difficulty just to obtain an adapted grid for the given PDE. We investigate whether relatively inexpensive adaptive grid methods, which use only *a priori* problem information, are useful. We find that such methods are feasible and we believe it is reasonable to assume the availability of sufficient *a priori* information in many cases. Our methods adapt by moving points in a rectangular *tensor product* grid; that is a grid consisting of the intersection of a set of grid lines in one coordinate direction with a set of lines in the other direction. After adaption, grids of this type are still logically tensor products. Besides simplifying the programming task, this class of grids seems to be the most promising with respect to parallelization (see [8]).

We discuss typical sources of *a priori* information in the next section. Section 3 describes two simple adaption procedures which are both effective and efficient for many problems. In Section 4 we discuss a computational framework for developing *a priori* grid adaption strategies. We conclude by illustrating the performance of the two methods of Section 3 with an example.

2 *A Priori* Problem Information

It is not uncommon in approximately solving PDEs to have some prior knowledge about a given problem, or about the behavior of the solution. This information may come from the physical system which is being modeled, or from similar problems with known or already computed solutions. It is common for example, that a PDE problem is identical to another problem except for one coefficient or a small change in the right side g. A scientist or engineer who has worked with many similar problems usually has an idea of where grid adaption my prove helpful.

Problem data may also provide valuable *a priori* information. By "data", we mean not only the coefficient functions and right side functions of the operator and boundary conditions, but the shape of the boundary itself. It may be that one expects a singularity in a coefficient to dominate the error in solving a problem, or the behavior of the right side g may give insight into where the problem is difficult. The behavior of one of these functions may be determined analytically or visually using a plot of the function. The shape of the boundary is another aspect of the problem data which can yield *a priori* information. For non-rectangular domains, a re-entrant corner or a highly irregular side may need to be handled with grid adaption.

A previous approximate solution to the problem can also be viewed as *a priori* information. Although technically *a posteriori* in nature, information based on an initial solution is obviously valuable and can be used as if it were known *a priori*. This is a reasonable approach if the initial solution uses a relatively coarse grid or an inexpensive method. Given this rough solution, one can

adapt the grid and re-solve the problem using a finer grid or a higher order method. Since the initial solution is inexpensive relative to the second one, the information used to adapt the grid is essentially *a priori*.

3 Two *A Priori* Grid Adaption Methods

We describe two methods which generate adapted grids based on *a priori* problem information. These methods are be both effective and efficient. They are effective in that they are good implementations of grid adaption, yielding considerable gains in accuracy over non-adapted, uniformly spaced grids. They are efficient in that the time taken to adapt the grid is very small with respect to the entire computation. Both methods rely on input indicating a particular area of the domain R which is thought to contain a difficulty; both use simple schemes to generate quickly a smooth adaptive grid.

3.1 The Point A Priori Method

The inputs to the *Point A Priori* grid adaption method are the coordinates of a point (x_0, y_0), the degree of adaption β, and the radius of adaption ρ. These parameters satisfy

$$\begin{aligned} &(x_0, y_0) \in R, \\ &\beta \geq 0, \\ &0 < \rho \leq 1. \end{aligned}$$

Given an initial uniform $n_1 \times n_2$ grid on $R = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, we move grid points inside a *circle of adaption* toward the specified point (x_0, y_0). The circle of adaption has center (x_0, y_0) and radius $r = \rho * d_{\max}$, where $d_{\max}$ is the maximum distance from (x_0, y_0) to a corner of the domain. The default value $\rho = 1$ means all grid points are moved. When $\rho < 1$ points outside the circle of adaption remain uniformly distributed.

Grid points inside the circle of adaption are moved in the x and y directions independently. We describe the adaption in x; movement in y is done in exactly the same way. For each row of grid points in the uniform grid we determine an interval of adaption $[a, b]$ as follows. Suppose the jth row lies along the line $y = y_j$. Let x_l and x_r be the x-coordinates of the two points where the line $y = y_j$ intersects the circle of adaption, assuming such intersections occur. Then

$$\begin{aligned} a &= \max(x_l, x_{\min}) \\ b &= \min(x_r, x_{\max}). \end{aligned}$$

Suppose m_1 of the uniformly spaced points lie to the left of a and m_2 lie to the right of b. These $m_1 + m_2$ points are not moved. The $n_1 - (m_1 + m_2)$ points in the adaption interval $[a, b]$ are chosen to equidistribute the integral of $e^{-\beta|x-x_0|}$ between a and b. In order to smoothly reduce the strength of the adaption as the distance from (x_0, y_0) increases, we replace β by

$$\beta' = \frac{r - |y_j - y_0|}{r} * \beta.$$

Thus, if

$$\int_a^b e^{-\beta'|x-x_0|}\, dx = C$$

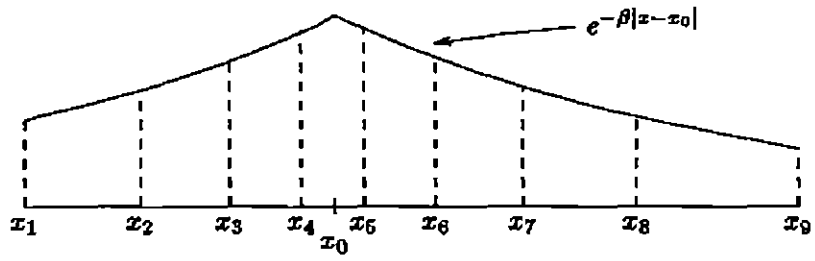

Figure 1: The equidistribution strategy of the Point A Priori method. Points $x_2, x_3, \ldots, x_8$ are chosen so that the area under the curve in each interval is constant.

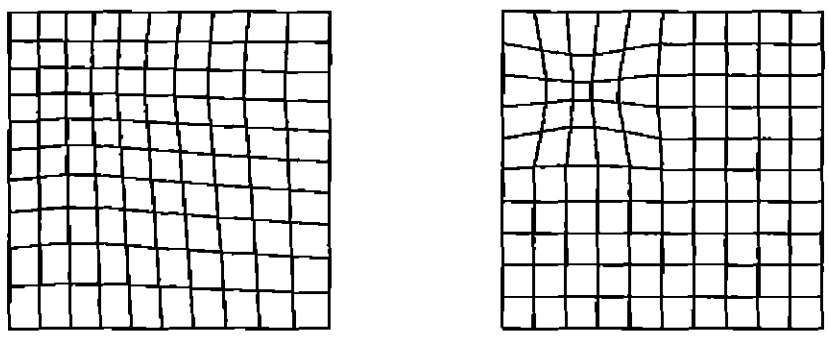

Figure 2: Sample adapted grids generated by Point A Priori. Parameter values (β, ρ) are (left) $(1, 1)$ and (right) $(10, .25)$. Input point (x_0, y_0) is $(.25, .75)$ for both grids.

for a constant C, we choose the x-coordinates $\{x_i : i = m_1 + 1, \ldots, n_1 - m_2\}$ by requiring

$$\int_{x_{i-1}}^{x_i} e^{-\beta'|x-x_0|}\, dx = \frac{C}{n_1 - (m_1 + m_2) - 1}.$$

The endpoints x_{m_1+1} and $x_{n_1-m_2}$ are fixed at a and b respectively.

Figure 1 illustrates the equidistribution strategy for a case with $n_1 = 9$ grid points and $m_1 = m_2 = 0$. Notice that increasing the degree of adaption β causes $e^{-\beta|x-x_0|}$ to rise more sharply near x_0, and relatively more points are located there. Figure 2 shows two typical adapted grids generated by the Point A Priori method. The input point (x_0, y_0) is $(.25, .75)$ in both cases. In the first grid the default values $\beta = 1$ and $\rho = 1$ are used. In the second grid the degree of adaption is increased to $\beta = 10$ and the radius reduced to $\rho = .25$.

3.2 The Curve A Priori Method

A second and somewhat more general *a priori* grid adaption method is *Curve A Priori*. Rather than specifying a single point toward which grid points are moved, this method moves points toward a curve along which the difficulty may lie. The curve is given parametrically. There are parameters β and ρ with meanings similar to those for the Point A Priori method. Optionally, adaption may be done in only one direction. This is useful in dealing with a boundary layer, for example.

In the general case, points in an initially uniform $n_1 \times n_2$ grid are moved in both the x and y directions. This is done in two main phases. The psuedo-code in Figure 3 summarizes the steps of the first phase. Given an initial grid $\{(x_{ij}, y_{ij}) : i = 1, \ldots, n_1; j = 1, \ldots, n_2\}$ uniform in x and y,

```
1.    for i = 1 to n_1 do
2.        call intersect (x_i, y*, α)
3.        if (α ≥ 0.2) then
4.            β' := (1/2 + α/π)β
5.            (y_i1, y_i2, ..., y_in_2) := equidistribute(y*, β')
6.        else
7.            for j = 1 to n_2 do
8.                y_ij := y_{i-1,j} + 1/2 (y_{i-1,j} - y_{i-2,j})
9.        endif
```

Figure 3: Psuedocode for first phase of Curve A Priori.

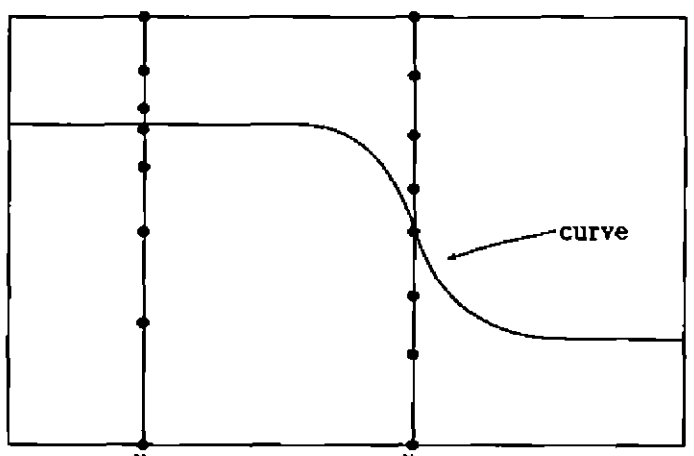

Figure 4: Two cases for the Curve A Priori method. Grid line x_i intersects the curve at a right angle so points are drawn strongly toward the intersection. Grid line x_j is nearly tangent so points are more evenly distributed.

new y-coordinates for each grid point are chosen by the steps shown. For each of the n_1 vertical lines $x = x_i$, we look for an intersection with the curve of difficulty (line 2 of Figure 3). If such a point (x_i, y^*) exists, we also estimate the angle α of the intersection. If no intersection is found, α is set to zero. Assuming an intersection is found, and assuming α is not too small, we proceed to set the y-coordinates of the grid points along the line $x = x_i$ (lines 4 and 5). This is done in a manner similar to the equidistribution step of the Point A Priori method. In particular, the n_2 values $\{y_{ij}\}_{j=1}^{n_2}$ are chosen so that

$$\int_{y_{i,j-1}}^{y_{i,j}} e^{-\beta'|y-y^*|}\, dy = C, \qquad \text{for } j = 2, \ldots, n_2,$$

where C is a constant. Notice that the value β', which governs the strength of the adaption, is less than β in proportion to how much the angle α falls short of $\pi/2$. When the angle of intersection is small we are more careful about concentrating points close to the intersection, since points may be pulled away from other difficult regions. If $\alpha = \pi/2$ then $\beta' = \beta$. Figure 4 shows how the angle α affects the strength of adaption. When the line and the curve are nearly tangent ($\alpha < 0.2$)

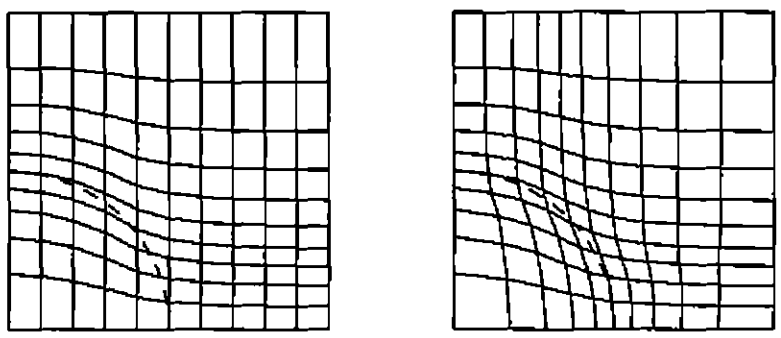

Figure 5: Example 11 × 11 grid after first (left) and second (right) phase of Curve A Priori. The dashed line is the curve of difficulty.

we do not use the equidistribution strategy at all. In that case (lines 7 and 8), the y-coordinates are chosen based on two neighboring columns of points. The jth horizontal grid "curve" is simply continued, with half the slope. The algorithm of Figure 3 is a simplification in several ways. In practice, the algorithm first sets the y-coordinates along all vertical grid lines which intersect the curve of difficulty at an angle $\alpha \geq 0.2$. Then the remaining points (if any) are chosen based on neighbors to the left or right, whichever is available. The current implementation allows a vertical grid line to intersect the given curve of difficulty at most once. This is not a general restriction of the method, although the equidistribution strategy would need to be modified if two or more intersections were allowed.

Figure 5 shows a typical mapping grid after the first phase of the Curve A Priori grid adaption method. Note how points are chosen along the vertical grid lines which do not intersect the curve. These points are adapted based on neighboring points that do intersect the curve.

The second phase of Curve A Priori is essentially a repeat of the first phase, with the points adapted horizontally rather than vertically. The points are moved along the horizontal "curves" determined by the first phase (see Figure 5), instead of strictly in the x direction. The criteria for adaption is the same as in Phase I. The angle of intersection (if any) of a given horizontal grid curve with the curve of difficulty determines the degree to which points are drawn toward the intersection. The points must move along the curve on which they lie. The equidistribution calculation is done in one dimension, and the results are mapped to the grid curve. If a grid curve fails to intersect the curve of difficulty, a smooth extrapolation from neighboring points is used. The second grid in Figure 5 is a typical grid after the second phase of the Curve A Priori algorithm.

4 Multidomain ELLPACK

The two *a priori* grid adaption methods described above are implemented as ADAPT modules in Multidomain ELLPACK (MDE). MDE is a problem solving system based on ELLPACK, a system for approximately solving elliptic PDEs [10]. With MDE more than one domain may be used to solve an elliptic problem. Just as in ELLPACK, MDE allows the user to state an elliptic PDE in a convenient very high level language, and then choose from a wide array of problem solving modules. Standard ELLPACK computes approximate solutions to elliptic problems in general two-dimensional domains or in three-dimensional boxes. For two-dimensional problems, MDE extends ELLPACK by allowing the use of more than one domain in the solution of a single problem. Mappings may be defined between these domains, and MDE handles the transformation of the

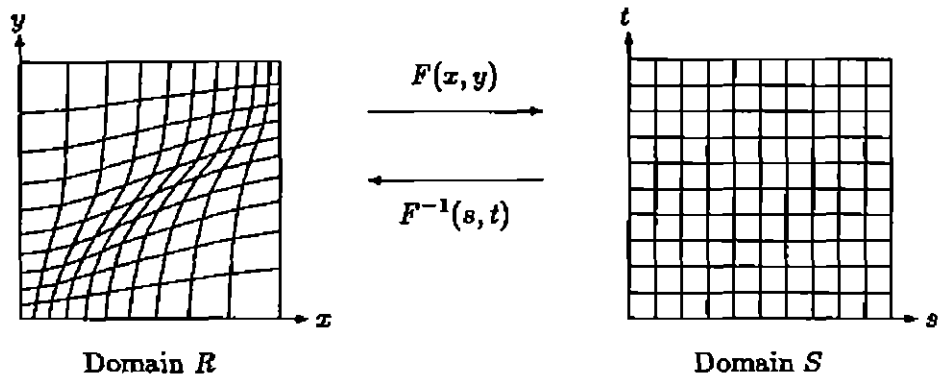

Figure 6: The two domains and grids used in adaptive grid domain mapping.

PDE automatically. MDE includes the menu-generation and interactive graphics capabilities of Interactive ELLPACK [5]. The interactive graphics features are particularly valuable in a multiple-domain environment, where changing geometries and grids are often most easily seen graphically.

Grid adaption fits easily into the multiple-domain framework of MDE. We view grid motion as a problem transformation or domain mapping. Conceptually, we use two domains to solve the problem (see Figure 6). The PDE is posed on a *problem domain* R. The problem is transformed via a mapping to the *solution domain* S. A function F maps points in R to points in S. A fixed uniform tensor product grid is used to discretize the transformed PDE on S. F^{-1} maps points in this uniform grid onto a curvilinear grid in R. The mapping is chosen so that this curvilinear grid is adapted, in some sense, to the given problem. Obviously, the task of choosing F (or F^{-1}) to optimize the distribution of points is a nontrivial one. Once it is done however, we can solve the transformed PDE in S by applying one of the problem solving modules in ELLPACK. Since the grid in Domain S is uniform, the potential for parallelism in the numerical solution may be more easily exploited as well, despite the use of grid adaption (see [8]).

Each of the ADAPT modules in MDE takes basically the same approach to determining the mapping F. Rather than searching directly for a suitable function F, we simply relocate the grid points in R according to some criterion. A mapping F^{-1} is then chosen which best approximates the new point distribution. We use a piecewise bicubic spline to represent each coordinate of F^{-1}. This choice provides enough flexibility to approximate a wide variety of point distributions while still maintaining the continuity needed to transform the problem smoothly. The coefficients of F^{-1} are determined by least squares. There are typically many more grid points in R than coefficients in F^{-1}. The least squares step smooths the mapping, in case the points were moved without much concern for smoothness. We find that the smoothness of the adaptive mapping is often crucial, particularly when high order discretization methods are used. A tensor product formulation of the bicubic splines is exploited to make the least squares step inexpensive. In practice, the grid used to select the adaptive mapping need not be as fine as the grid used in discretizing the problem; it need only be fine enough to allow an effective mapping to be constructed.

Given the mapping F^{-1}, the transformed problem is solved in Domain S. First and second order partial derivatives of F^{-1} are needed to compute the coefficients of the transformed problem. If F must be evaluated at some point (x, y) in R, we use a special two dimensional secant method to invert F^{-1} numerically. Evaluations of F are not needed to discretize the transformed PDE,

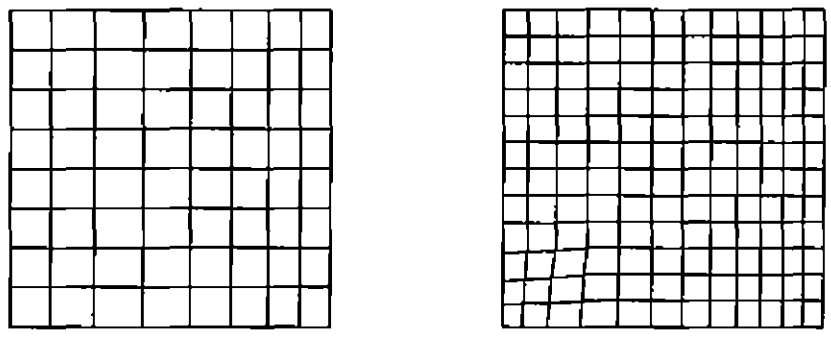

Figure 7: Adapted 9×9 grid generated by Curve A Priori with $\beta = 1$ and $\rho = 0.8$ (left). Adapted 13×13 grid generated by Point A Priori with $\beta = 2$ and $\rho = 0.25$ (right).

but only when solution values on Domain R are desired. The numerical inversion step converges rapidly since a good initial guess is normally available.

5 An Example

We give an example to illustrate the effectiveness of the Point and Curve A Priori grid adaption schemes. This example demonstrates how these methods may be combined to produce grids which are adapted to more than one area of difficulty. Obviously, the greater the number and complexity of difficult regions, the harder it is to produce "optimal" grids.

Consider a Poisson problem on the unit square with Dirichlet boundary conditions. The right side g is chosen so that the problem has true solution

$$\mathit{true}(x,y) = \mathit{smooth}(x,y) + \mathit{singpt}(x,y) + \mathit{blayer}(x,y), \tag{3}$$

where

$$\begin{aligned} \mathit{smooth}(x,y) &= [\cos(y) + \sin(x-y)] * [1 + \sin(x/2)] \\ \mathit{singpt}(x,y) &= \left(\sqrt{x^2+y^2}\right)^{1.5} \\ \mathit{blayer}(x,y) &= \exp[-10 * (1-x)]\,. \end{aligned}$$

The true solution (3) is from the set of parameterized PDE solutions in [9]. We use ELLPACK modules INTERIOR COLLOCATION (collocation with Hermite bicubics) and BAND GE (band Gauss elimination) to approximate the solution to this problem. The computations are done in double precision using the f77 FORTRAN compiler on a Ridge 32 under ROS 3.3. The error in solving this problem is greatest along the right boundary and near the origin (see first plot in Figure 9). Assuming *a priori* information to this effect, it is easy to generate an adapted grid with Point and Curve A Priori.

We first apply the Curve A Priori method to construct a mapping adapted toward the boundary layer. Figure 7 (left) shows an adapted 9×9 grid after this step. We specify radius of adaption $\rho = 0.8$ so points near the origin are not moved. Adaption is only done in the x direction. The time to generate this adaptive mapping is .16 seconds.

Table 1: Time in seconds and error in solving original problem (Domain R) and transformed problem (Domain S).

	Solution in R		Solution in S	
Grid	Time	Error	Time	Error
7	3.36	9.86E-3	4.30	4.58E-3
10	10.95	2.48E-3	12.80	1.12E-3
13	27.50	8.26E-4	30.78	3.95E-4
19	109.30	2.70E-4	116.31	9.03E-5
25	304.38	1.23E-4	317.36	2.54E-5

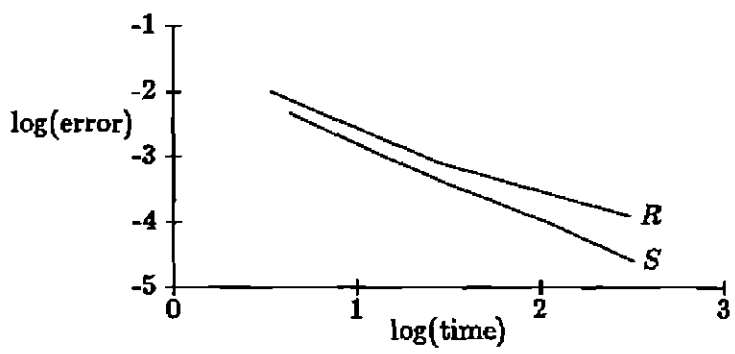

Figure 8: Log-log plot of error versus time for solution on Domains R and S.

We apply Point A Priori adaption to improve the solution near the origin. The second grid in Figure 7 is the result of composing the two adaptive mapping methods—Curve A Priori followed by Point A Priori. When there is more than one area of adaption, the dimension of the the grid used to construct the mapping must sometimes be increased. More spline pieces may also be needed in the representation of the mapping F^{-1}. For this example we construct the mapping based on a 13×13 grid, and represent F^{-1} with three spline pieces in each direction (the default is two). The time taken by Point A Priori is .47 seconds.

Having constructed the adaptive mapping F^{-1} from the second grid in Figure 7, MDE allows us to solve the transformed problem on Domain S. Table 1 summarizes the results for solutions on both domains. We use the maximum absolute error over a uniform 40×40 grid in Domain R to estimate the error. Solving the transformed problem takes slightly longer than the original problem for a given grid size n. This is expected since the coefficients of the transformed PDE are more expensive to evaluate than those of the original problem. Since the time to solve the linear system is constant for fixed n however, and since as n grows this time dominates the computation as a whole, the relative difference in time between solving in R and solving in S decreases as n increases. For all grid sizes the accuracy of the solution in S is significantly better. With a 25×25 grid (2304 unknowns) there is nearly an order of magnitude improvement in accuracy and only a 4% increase in time. The plot of time versus error in Figure 8 shows the considerable improvements gained by the grid adaption. The second contour plot in Figure 9 shows that the error is reduced an "spread out" in both of the difficult areas.

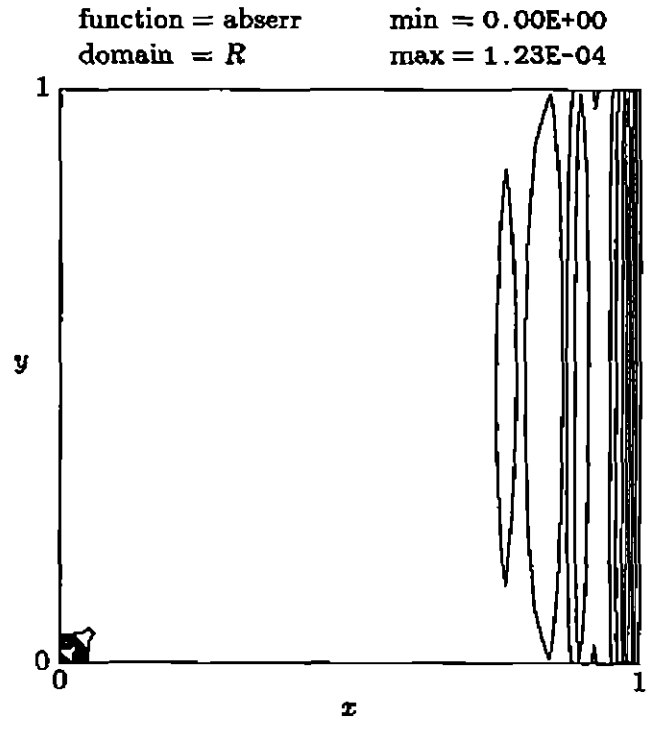

Figure 9: Error in solving original (top) and transformed (bottom) problem using 25×25 grid. Error is measured on a 40×40 grid in Domain R.

References

[1] S. Adjerid and J. E. Flaherty, "A moving finite element method with error estimation and refinement for one-dimensional time dependent PDEs", *SIAM J. Numer. Anal.*, 23(1986), pp 778–796.

[2] S. Adjerid and J. E. Flaherty, "Adaptive finite element methods for parabolic systems in one and two space dimensions", Technical Report 86-20, Rensselaer Polytechnic Institute, 1986.

[3] I. Babushka, J. E. Flaherty and J. Chandra (eds.), *Adaptive Computational Methods for PDEs*, SIAM, Philadelphia, 1983.

[4] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic PDEs", Technical Report NA-83-02, Stanford University, 1983.

[5] W. R. Dyksen and C. J. Ribbens, "An interactive problem solving environment for elliptic partial differential equations", Purdue University, Computer Science Department Report CSD-TR 588, 1986.

[6] J. E. Flaherty, M. Coyle, R. Ludwig and S. F. Davis, "Adaptive finite element methods for parabolic partial differential equations", in *Adaptive Computational Methods for PDEs* (I. Babushka, J. E. Flaherty and J. Chandra, eds.), SIAM, Philadelphia, 1983, pp 144–164.

[7] K. Miller and R. N. Miller, "Moving finite elements I", *SIAM Jnl. Num. Anal.*, 18(1981), pp 1019–1032.

[8] C. J. Ribbens, Ph.D. dissertation, Department of Computer Sciences, Purdue University, 1986 (unpublished).

[9] C. J. Ribbens and J. R. Rice, "Realistic PDE solutions for non-rectangular domains", Purdue University, Computer Science Department Report CSD-TR 639, 1986.

[10] J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[11] J. F. Thompson, "A survey of dynamically-adaptive grids in the numerical solution of partial differential equations", AIAA Technical Report 84-1606, 1984.

[12] J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, *Numerical Grid Generation*, North Holland, New York, 1985.